Monitoring the I2P network

Juan Pablo Timpanaro, Chrisment Isabelle, Festor Olivier

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Monitoring the I2P network

Juan Pablo Timpanaro, Isabelle Chrisment, Olivier Festor

N° 7844 — version 1

initial version October 2011 — revised version Décembre 2011

Domaine 2

ISSN 0249-6399 ISRN INRIA/RR--7844--FR+ENG

INSTITUT NATIONAL
DE RECHERCHE
EN INFORMATIQUE
ET EN AUTOMATIQUE

INRIA

centre de recherche
NANCY - GRAND EST

Monitoring the I2P network

Juan Pablo Timpanaro, Isabelle Chrisment, Olivier Festor

Domaine : Algorithmique, programmation, logiciels et architectures
Équipe-Projet madynes

Rapport de recherche n° 7844 — version 1 — initial version October 2011 — revised version Décembre 2011 — 15 pages

Abstract: We present the first monitoring study aiming to characterize the usage of the I2P network, a low-latency anonymous network based on garlic routing. We design a distributed monitoring architecture for the I2P network and show through a one week long experiment the ability of the system identify a significant number of all running applications, among web servers and file-sharing clients. Additionally, we identify that 37% of published I2P applications, which turn out to be unreachable after their publication on the I2P distributed database.

Key-words: I2P, Anonymous Networks, Monitoring

Centre de recherche INRIA Nancy – Grand Est
LORIA, Technopôle de Nancy-Brabois, Campus scientifique,
615, rue du Jardin Botanique, BP 101, 54602 Villers-Lès-Nancy
Téléphone : +33 3 83 59 30 00 — Télécopie : +33 3 83 27 83 19

Monitoring the I2P network

Résumé : Pas de résumé

Mots-clés : I2P, Anonymous Networks, Monitoring

1 Introduction

Over the last decade, anonymous communications have gained more and more interest. These networks offer an excellent support for preserving users anonymity and privacy against eavesdroppers or third-party sniffing; they are also used for illegal activities, i.e. copyrighted file-sharing.

As with the normal Internet, in an anonymous network a message is routed through a set of intermediate nodes, before it reaches its destination. However the goal of most anonymous networks is to hide the originator of the message, so the final destination will only see the last intermediate node as originator and not the real initiator of the message. Most of the current anonynous networks use a variant of onion routing [4] and layered encryption for achieving their goal.

These networks are classified in two types: *high-latency* and *low-latency* networks.

Most of nowadays high-latency networks are based on the concept of *mix*[2], which can be defined as a process which accepts messages, groups them into a batch and forwards them later on, based on a given batching algorithm. Since these algorithms introduce delays in the routing of a message, these kind of networks are only suitable for non-interactive applications, like e-mail. The *Mixmaster*[12] network is one of such available networks.

On the other hand, most of applications require a fully interactive connection among users, such as web-browsing or chatting. Low-latency anonymous networks reduce delays and bandwidth usage to improve the speed of the network, although these approaches weaken the anonymity provided by the network itself.

The I2P[7] network, a low-latency message-oriented anonymous network was mainly designed to allow a fully anonymous conversation between two parties within the I2P network limits. Since out-proxying traffic is not the goal of the network, the number of out-proxies in I2P is limited, on the contrary to the Tor network[14], for example.

I2P contains a full range of available applications: anonymous web-browsing, chatting, file-sharing, web-hosting, e-mail and blogging among others. Except for anonymous web-browsing that necessarily requires an out-proxy to the normal Internet, all other listed applications interact between each other within the I2P network boundaries.

We make the following contributions: 1) we present the first application-layer monitoring of I2P, 2) we propose a fully operational architecture for monitoring *destinations* on I2P, 3) we provide, over three one-week measurement experiments of the I2P network, a high-level view of the network and identify which applications are most used and when they are used, on the contrary to statistics web sites[1], which only give the number of applications, but not the type.

This paper is organized as follows. Section 2 introduces the I2P network and its main components and features, including peer profiling and its distributed database. Section 3 describes our monitoring architecture. We detail how we use I2P's distributed database to collect data and how we process this data to characterize the network. In Section 4 an initial experiment is presented, covering three full weeks (non-consecutive) of monitoring. Section 5 points out previous and current work on low-latency anonymous networks, mainly I2P

[1] Such as `http://stats.i2p.to/`

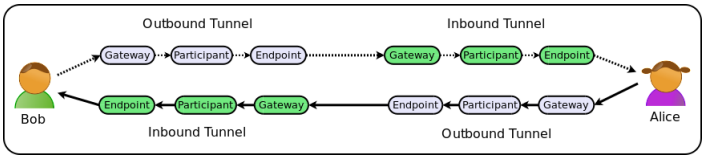


Figure 1: Simple tunnel-oriented communication in I2P

and Tor, ranging from attacks development to network monitoring. Section 6 concludes the paper.

2 The I2P network

The I2P network allows anonymous communications between two parties through an abstraction layer, which uncouples the association between the user and its identity. Basically, an application using I2P will not longer be reachable through an IP, but through an location independent identifier. The following sub-sections address the I2P network and its features.

2.1 Overview

The network is formed of a group of *routers*. A router runs the software that allows any application to communicate through I2P. Applications running on top of it will have a *destination* associated, which receives incoming connections from third parties. The secret lies in which destination is associated to which router and not in the fact that a user is running an instance of the router. This uncoupling between the router and the destination provides a certain degree of anonymity.

I2P uses an extension of the well-known onion routing approach [4], in which a message is routed from its originator to the final endpoint through several intermediate nodes using layered encryption. The originator adds to the message to be sent an encryption layer for every node in the path, each intermediate node *peels off* one of these layers, exposing routing instructions along with still-encrypted payload data, and finally the last node removes the final layer of encryption, exposing the original message to the endpoint.

This extension is called garlic routing, and allows to the originator to include several messages in a single *onion*. I2P currently uses this approach to include the return destination for a given message as well as status messages.

The path through a selected list of nodes is called a *tunnel* and represents a key concept in I2P.

2.2 Tunnels

Every tunnel is unidirectional, and is formed by the gateway (entry point), a set of participants (intermediate nodes) and an endpoint. Additionally, two types of tunnels exist. *Inbound* tunnels allow an user to receive data, and *outbound* tunnels to send data. A fully bidirectional communication between two users will involve four tunnels, one inbound and one outbound for each user.

Figure 1 illustrates a communication between Alice and Bob. First, let's assume that Alice and Bob know each other destinations. Alice will send a

message through her outbound tunnel, targeting one of Bob's inbound tunnel gateway. Once the message reaches the *gateway*, which is the entry point for Bob's tunnel, it will be forwarded all the way through Bob's router. Alice does not have any knowledge about Bob's inbound tunnel, but only about the entry point: Bob might have as well a tunnel composed of 1 or 100 intermediate nodes, but Alice ignores this. This property is the earlier mentioned *decoupling*. The same procedure is used by Bob when sending data to Alice, he will send a message targeting Alice's inbound gateway.

2.3 Profiling algorithm

Tunnels are created every 10 minutes, and then dropped. This feature complicates a traffic analysis attack, since a 10 minutes time window is rather small to acquire any knowledge of a network[1]. Therefore, every 10 minutes, a user selects new nodes for tunnels, which are selected among all the routers in the network.

The easiest way would be to choose random routers among the fastest ones in the network, and place them in a random order through the tunnel. However, I2P's anonymity and performance depend on not being random in this aspect. I2P leans on a constantly local profiling of all seen routers, to characterize every peer regarding its performance, latency, and availability. A four-tier scheme is used to classify routers: fast and high capacity, high capacity, not failing, and failing.

Peers are profiled based on their interactions with the profiling peer. Among the attributes used for profiling we can find success rate of lookups in the network database, success rate of message through a tunnel containing the profiled peers, the number of peers a profiled peer can introduce to us. All kind of indirected behaviour is recorded and used for profiling, to the contrary of claimed performance from routers. No published performance information is used in local profiling, so as to avoid simple attacks.

2.4 I2P netDB

The *netDB* is one key concept in the I2P network. It is a distributed hash table based on the Kademlia [10] protocol, used to store and share network metadata. However, contrary to the Kademlia protocol, all the ppers in the I2P network are not part of the DHT, but only those fast I2P users, the so called *floodfill* peers. Any user with high bandwidth can appoint itself as a floodfill peer, if the number of floodfill peers in the network decreases for some reason.

There are two types of network metadata, *leasesets* and *router infos*. A leaseset provides information about a specific destination, like a web server, a BitTorrent client, an e-mail server, etc. A router info provides information about a specific router and how to contact it, including the router identity (keys and a certificate), the address where to contact it (ip and port), several text options and a signature.

A leaseset gives a number of entry points for a client destination. When Alice creates a destination (for an e-mail server, for example), a set of *leases* are created and grouped into a leaseset. Each lease contains the tunnel gateway router (where Bob should send the messages), when the tunnel expires (deadline for using this gateway) and a tunnel ID.

This distributed database contains an extra security feature, to harden a localized *Sybil attack*. The key used to index a record in the netDB is computed as `HASH(ID + date)`, in which the *ID* is the record ID, and *date* is the current date. A record ID remains fix as long as the record is conserved, however the record indexing key will change every day.

It means that at midnight, all indexing keys will be changed and therefore re-published in other locations in the DHT. Even though some queries might fail around midnight, this approach avoids an attacker to launch a simply localized Sybil attack, since the attacker will have to re-compute its Sybils IDs using a key-to-key dictionary. A deeper view of this network database is out of the scope of this document, however a further insight of the netDB and the flooding mechanism can be found in the official I2P website [7].

2.5 Differences with the Tor network

We could dedicate an entire section for specifying differences between these two networks, however we will only focus on high-level differences. Both Tor and I2P are low-latency networks, although Tor is intended for out-proxying traffic to the regular Internet, while I2P has hardly out-proxies. However, one of the main difference is how both networks manage the participants. Tor has a central server directory, which provides an overall view of the network and eases statistics retrieval. On the other hand, I2P is based on a distributed Kademlia-based database, along with a peer profiling algorithm for peer selection. This distributed component hardens the network and makes it more resilient to shut-down attempts, on the contrary to the Tor central-based directory.

3 I2P monitoring architecture

We aim to monitor the I2P network, and determinate which are the most used applications running on top of it, so as to characterize the usage of the network. Is I2P mostly used for file-sharing? Or is it mostly used for anonymous web hosting? Can we even characterize the usage of the network?

We focus on two main applications, web servers (anonymous web hosting) and file-sharing clients. For file-sharing clients, we will consider I2PSnark, an built-in BitTorrent client in I2P.

So as to achieve this goal, we could inquire every single user in the network for the running application(s) at a given moment. However, as earlier mentioned, I2P bases its anonymity on de-coupling the identity of an user (provided by its *router info*) from the application it is running (provided by its *leaseset*). Therefore, the challenge is to determine which application is running in a given *leaseset*, even if we can not link an user with its running applications.

3.1 I2P and its netDB

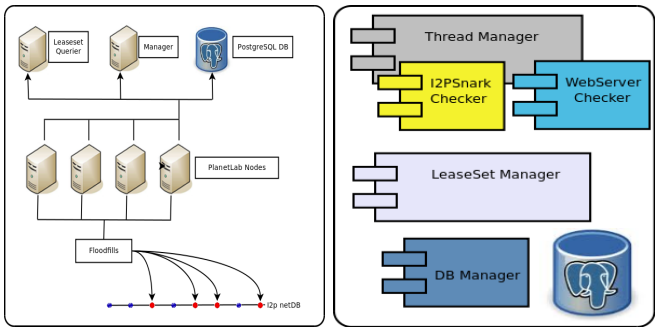
Since the netDB contains, normally, all the *router infos* for every router in the network, and all the published *leasesets*, it becomes a key component in our monitoring architecture. We try to retrieve from the netDB as many *leasesets* as possible, and query them. To collect *router infos* and *leasesets*, we place a small number of *floodfills* in the distributed hash table. As mentioned before,

floodfill nodes have the capacity to store and index this information, and they are a sub-set of all the nodes in the network.

We take advantage of the mechanism to become a floodfill in the I2P network, in which any router can postulate itself as a floodfill router. Once our routers announce themselves as floodfills, they will start to receive store requests and provide to others with this information.

3.2 Monitoring Architecture

Our monitoring architecture is divided in two main parts.



(a) Leasesets retrieval (b) Leaseset querier

Figure 2: Complete monitoring architecture

Figure 2(a) presents the approach used to collect *leasesets*. Each floodfill is running in one PlanetLab node, and logs every received request to a PostgreSQL server, located in our high security lab.

Figure 2(b) shows the architecture of the *leaseset querier*, which runs in parallel with the architecture showed in figure 2(a). The *Leaseset Manager* periodically retrieves new leasesets from the database, while the *Thread Manager* creates different threads to test a given destination for a particular application (a Web Server or an I2PSnark client). Once a leaseset is tested, the result is kept in the PostgreSQL server. This architecture needs a running I2P router, so as to join the I2P network and communicate with different leasesets.

3.3 Analysis of I2P applications

In a first series of experiment, we analyse two kinds of applications running on top of an I2P router, a web server and an I2PSnark client, which is a modified bittorrent client designed to run on top of I2P.

On the one hand, to tag a given leaseset as a web server, we open an I2P through it and send a `GET` message. If the response contains well-known http keywords, then that leaseset corresponds to a web server.

On the other hand, the approach used to test an I2PSnark client is shown in figure 3, which we obtained by analysing the I2PSnark code, along with basic black-box testing. We consider the following behaviour of an I2PSnark client before detailing our approach. If an I2PSnark client receives a malformed BitTorrent message, it does not respond. When it receives a well-formed BitTorrent message requesting a torrent that the client is not currently sharing, it closes the I2P connection immediately.

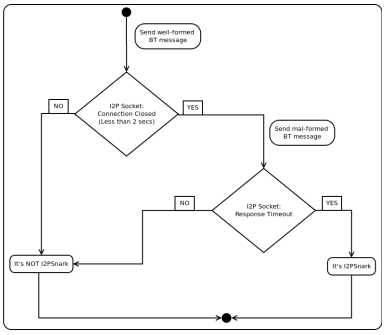


Figure 3: Algorithm for testing an I2PSnark client

With this slight knowledge, we proposed the following approach to determinate if a given leaseset corresponds to an I2PSnark client. First we open an I2P socket through a given leaseset using the underlying I2P router. Once the connection is established, we send a first message, a well-formed BitTorrent message, requesting a random torrent. If that given leaseset is actually running an I2PSnark client and not sharing the torrent, it will immediately close the I2P socket. Second, we re-open the connection and send a malformed BitTorrent message. If the response timeouts, then we conclude that the given leaseset is running an I2PSnark client. If we do not receive any answer for the first message, we can not assume anything, and we will keep trying that leaseset with different applications.

4 Experiments

In this section, we present our experiments on monitoring the I2P network with the previously presented architecture, and then we detail a set of results.

4.1 Set-up

First we need to know the number of floodfills we have to place in the netDB. We consider the current number of floodfills in the network, which is estimated by our set of floodfill routers. We have, in average, 200 floodfills and considering the 8X replica set[2] for the netDB we should have a $200/8 = 25$ perfectly distributed floodfills to have a 100% coverage of the DHT space.

During this first series of experiments we used the Planet Lab test-bed, which has restrictions regarding bandwidth rates. We were able to place only 15 floodfills with minimum bandwidth settings, but still allowing us to perform as floodfills nodes. 15 floodfills give us around 60% of coverage of the DHT space if the floodfills are perfectly distributed, but since they are randomly placed, we got a bit less than 60%.

Each of our floodfills ran in one planet-lab node, and logged every `store leaseset` request. Our leaseset querier ran in parallel, analysing new lasesets and storing the results. We conducted 3 different one-week experiments, starting on September 12th, September 27th and October 4th.

[2]The *replica set* is the number of nodes in which a router will store its requests, since storing a request in a single node is not safe against DHT churn.

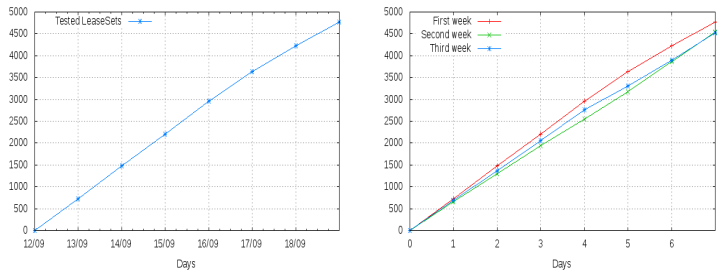
We logged four possible outcomes for a given leaseset:

- WebServer
- I2PSnark client
- Unknown: *The given leaseset is not running a web server nor an I2PSnark client*
- Destination not found: *The given leaseset is unreachable*

4.2 Experiments results

We show the results for the experiment ran on September 12th, although the three of them present similar results. Figure 4(a) shows the accumulative value of tested leasesets every day. We tested, in average, 676 new leasesets per day, at a constant rate of 28 leaseset per hour.

Figure 4(b) presents the number of leasesets tested every day for each of the three experiments, in which it can observed that on the three experiments, we tested the same amount of leasesets per day.

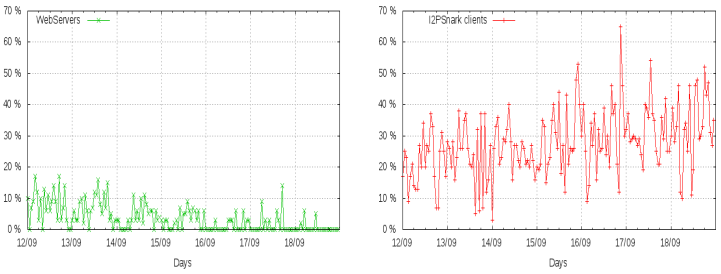


(a) Leasesets tested on September 12th - 19th

(b) Leasesets tested in every week

Figure 4: Leasesets results

Figure 5 gives the percentage of leasesets that we were able to identify and which we tagged for the first experiment. In average, we were able to identify 32.06% of all the leasesets queried, which most of them were I2PSnark clients, as shown in figure 5(b).



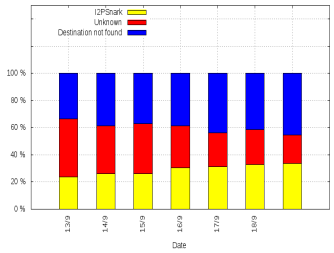
(a) Anonymous web servers

(b) I2PSnark clients

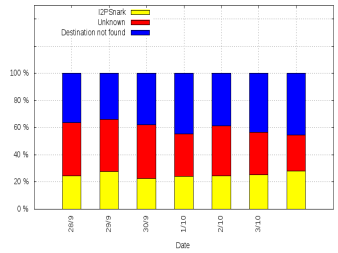
Figure 5: Identifiable leasesets (September 12th - 19th)

Since we consider only new leasesets every day, it is normal that the amount of known anonymous web servers decreases every day. Anonymous web servers do not change their published destinations, and therefore the amount of new anonymous web servers every day decreases through time as we maintain a history of already seen destinations. On the contrary, I2PSnark clients maintain a stable rate during the week, increasing during the weekend.

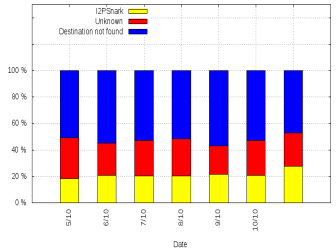
Figure 6 presents the full results in percentage of each weekly experiments, in which the *unknown* destinations along with the *destinations not found* values are shown.



(a) One week results (12/09 - 27/09)



(b) One week results (27/09 - 04/10)



(c) One week results (04/10 - 11/10)

Figure 6: Results for every one-week experiment

Table 7 summarizes the numbers of I2P destinations we analyzed for every experiment. For our last experiment, the *Destination not found* value is higher than we have seen in the previous experiments. The statistics web page of I2P[3] reports that between the 02/10 and the 09/10 the number of routers in the network increased considerably (from 7000 routes to 10000 routers, in average). This increase of new participants in the network could explain why we have a higher number of unreachable destinations, since many I2P users might have trying the network and testing different applications.

	Anonymous web servers	I2PSnark clients	Unknown	Destination not found	Total
First experiment	193	1312	1423	1793	4721
Second experiment	172	1106	1503	1757	4538
Third experiment	175	1057	1186	2349	4767

Figure 7: Results for every one-week experiment

[3] `http://stats.i2p.to/`

We have, on average, 30.16% of *unknown* applications, which means that we successfully opened a socket through these leasesets, but we failed at tagging them.

4.3 I2PSnark clients analysis

As we mentioned before, new leasesets were considered in our experiments. An I2PSnark client creates a new destination every time it starts, and therefore we have a stable rate of new I2PSnark clients in our results. We can also observe that most of them were identified during European night time. Figure 8 shows an increase of I2PSnark clients starting at 20:00 and a decrease around 07:00 AM. Moreover, during the week days, we have an average of 25.5% I2PSnark clients, and during the weekend this value climbed to 32.5%, indicating that I2P users dedicate more time for file-sharing during weekends, which is not surprising. Additionally, there are no time ranges during the weekend, since we had different pikes in our analysis during day time as well.

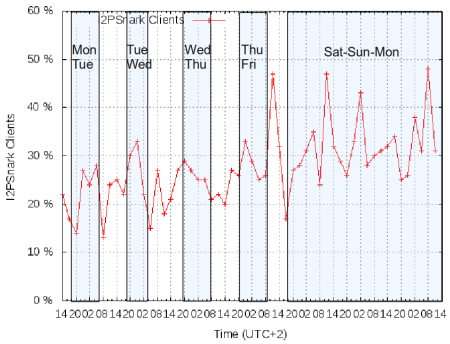


Figure 8: I2PSnark clients during one week (September 12th - 19th) / 3-hours interval

4.4 Anonymous web servers analysis

Since destinations for anonymous web servers or eepsites do not change over time, we enquire them every hour, to check their availability over time. Figure 9 shows the cumulative value of anonymous web servers online for a two-week period (28/09 - 11/10).

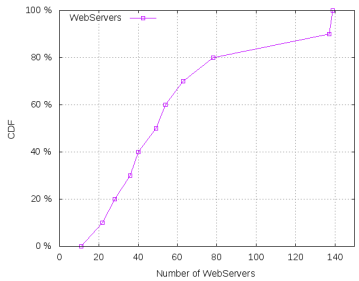


Figure 9: CDF of online anonymous web serves

We analysed 139 eepsites, in which we observe that more than 50% of them were online 70% of the total time and only two of them were only during the whole two week experiment.

Our monitoring approach allow us to detect new anonymous web servers as soon as they are published in the netDB, on the contrary to statistics web sites such as `perv.i2p` or `inproxy.tino.i2p`. These web sites scrape the list of known anonymous hosts[4] and present the uptime of every eepsite; if an anonymous web server is not published in the list, these web sites will not contact it.

4.5 *Destination not found* analysis

During our experiment, we had an amount of leasesets that were not available after being stored in the netDB by our floodfills, and as a result these were tagged as *Destination not found*. Despite every leaseset was double-checked after the first *destination not found* result, we kept on getting these status messages from our local router. In average, we got 37.70% of leasesets in which no answer was obtained while trying to open a socket through them for the first experiment. We ran the same experiment 3 times and we got an overall average of 38.42%.

We consider a set of possible problems for this situation. Firstly, a particular application might be publishing a set of destinations and only using a sub-set of destination, discarding the rest. Since there are a considerable set of available applications, it is non-viable to test every single application.

On the other hand, we focused on the file-sharing applications. We retrieved 2200 active torrents from the postman tracker[5] of I2P. For each torrent we got a list of destinations (which represents a torrent client) sharing that particular torrent, and we repeated this procedure every hour, so as to get as many new destinations for a given torrent as possible.

We cross checked all the retrieved destinations from the tracker along with the destination not found of our previously analysis. We got that 15% of the destination not found were from file-sharers using the postman tracker. This indicates that 85% of file-shares behave adequately in the I2P network. File-sharers do not start an I2P BitTorrent client for a short period of time, but they maintain the client connected longer.

We chose the postman tracker because it is the most stable tracker in I2P. Alternative trackers such as `tracker.welterde.i2p` and `tracker.rus.i2p` do not seem to be online most of the time, at least for the duration of our experiment.

5 Related work

Anonymous low-latency networks are gaining more and more interest. The *Tor* network is probably the one that is receiving most of the academic attention.

[4] Anonymous web servers are listed in a file, which is shared by every user in the I2P network. This file is known as the *hosts* file and along with a synchronization service in I2P, it provides a DNS-like service.

[5] `tracker2.postmam.i2p`

There are a series of studies regarding attacks on the Tor network, from timing attacks [6] to discovering *hidden services* from their clock skew [13].

Regarding monitoring of the Tor network, McCoy et al.[11] describe a monitoring approach, which takes advantage of being a Tor *exit node*, and analyses the application layer of outgoing traffic to determinate the protocol distribution in the network. They discover that most of the connections through Tor are interactive http traffic, while a few of them are for BitTorrent traffic. However, these few connections (3.33% according to their study) consume a disproportionally amount of bandwidth (40.20% of the total measured).

The study in [11] resembles what we want to achieve for I2P, however, the data collection methodology applied by McCoy et al. can not be used in I2P. Services in I2P interact within the network limits, and there is not out-going traffic for file-sharing or e-mail, for example.

More recently, Loesing et al. [9] presented a measurement of sensitive data on the Tor network, such as country of connections and outbound traffic by port. Additionally, Loesing [8] measured the *trends* of the Tor network from directory information. However, this network has a central component, a *directory server* and hence the monitoring approaches can not be applied to the I2P network, which does not include any kind of central component. The Tor metrics website, `http://metrics.torproject.org`, provides further technical reports on measurement of network components, such as relays and bridges.

In [16] an analysis of the peer profiling algorithm in I2P is presented, discussing its strengths and weaknesses together with future improvements. The authors conclude that I2P's profiling algorithms behave accurately when finding faster peers, increasing its performance when the I2P router requires more bandwidth. They also conclude that these algorithms do not provide an easy way of *tuning* the trade-off between anonymity and performance, although a user can tweak the length of a tunnel to make its connection either faster or more anonymous.

Crenshaw[3] studied how to identify I2P's hidden services. He successfully linked an anonymous website, known as *eepsite*, to its real Internet address. He takes advantage of the lack of control in the application layer, errors and mistakes that system administrators as well as developers made during the implementation and configuration of a web server. Even if it is not an I2P problem, he shows how important it is to properly set-up any application running on top of an anonymous layer.

Herrmann et al.[5] conducted an attack on the I2P network, so as to determining the identity of peers hosting anonymous web sites. They proposed a three-step attack, in which an adversary with modest resources will 1) get an estimated view of the victim's network, 2) attack the victim's fastest peers tier[6], so as to replace them with the attacker's peers, and finally 3) tag a given router as the host of an eepsite by using a traffic analysis technique developed by their own, based on statical patterns of http requests induced by the adversary. They conclude that churn in the fast and high capacity tier might be the main problem, and they propose solutions to avoid such a churn or reduce it.

Our work gives the I2P community a bird's eye view about the network usage, from an application point of view. We are taking the monitoring of this network one step further with our work, not by providing well-known statistics,

[6]Peer tiers are explained in section 2.

such as the number of routers or bandwidth usage[7], but by identifying running services on top of the I2P network.

6 Conclusion

We have developed the first high-level monitoring architecture for I2P, aiming to characterize the use of the network in term of available *destinations* and applications running on top of it.

After the first one-week experiment we analysed 4721 leasesets. We were able to identify 193 web servers and 1312 I2PSnark clients, and we did determinate that 37% of the published leasesets were off-line after their publication on the netDB. On the other hand, we were not able to identify 30% of the entire set of leasesets, which means that 30% of the network is not running a web server nor an I2PSnark client.

We also made an uptime analysis of anonymous web servers in I2P. We could observed that more than 50% of seen anonymous web servers were online 70% of the total time of the two weeks analysis, which indicates that most of I2P services remains available most of the time.

Ongoing work includes extending the approach to improve the qualitative identification of anonymous web servers and I2PSnark clients, as well as testing new I2P applications, such as IMule and I2Phex, both file-sharing applications. Including new torrent trackers in our analysis for unreachable destination is another major goal.

Future work will include a long-term measurement of the network. Even though a one week experiment might give us a glimpse about the current usage of the network, a monthly or even a longer measurement experiment may provide us with a deeper characterization of I2P. Additionally, we will deploy our monitoring architecture permanently, as well as developing a front-end to have a real time view of the network, based on the data collected by our architecture.

Acknowledgment: We thank the anonymous leading developer of I2P for his useful comments and reviews of this paper.

References

[1] K. Bauer, D. McCoy, D. Grunwald, T. Kohno, and D. Sicker. Low-resource routing attacks against anonymous systems. In *Proceedings of the 2007 Workshop on Privacy in the Electronic Society (WPES)*. Citeseer, 2007.

[2] David Chaum, Communications Of The Acm, R. Rivest, and David L. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24:84–88, 1981.

[3] Adrian Crenshaw. Darknets and hidden servers: Identifying the true ip/network identity of i2p service hosts. In *Black Hat DC 2011*, DC, 03 2011.

[7] Which can be found in `http://stats.i2p.to/`.

[4] D. Goldschlag, M. Reed, and P. Syverson. Hiding routing information. In *Information Hiding*, pages 137–150. Springer, 1996.

[5] M. Herrmann and C. Grothoff. Privacy-implications of performance-based peer selection by onion-routers: A real-world case study using i2p. In *Privacy Enhancing Technologies Symposium (PETS 2011)*, 2011.

[6] N. Hopper, E.Y. Vasserman, and E. Chan-Tin. How much anonymity does network latency leak? *ACM Transactions on Information and System Security (TISSEC)*, 13(2):13, 2010.

[7] I2P. The i2p netowrk. `http://www.i2p2.de/`.

[8] K. Loesing. Measuring the tor network from public directory information. *Proc. HotPETS, Seattle, WA, August*, 2009.

[9] K. Loesing, S. Murdoch, and R. Dingledine. A case study on measuring statistical data in the tor anonymity network. *Financial Cryptography and Data Security*, pages 203–215, 2010.

[10] P. Maymounkov and D. Mazieres. Kademlia: A peer-to-peer information system based on the xor metric. *Peer-to-Peer Systems*, pages 53–65, 2002.

[11] Damon Mccoy, Tadayoshi Kohno, and Douglas Sicker. Shining light in dark places: Understanding the tor network. In *In Proceedings of the 8th Privacy Enhancing Technologies Symposium*, 2008.

[12] Mixmaster. The mixmaster network. `http://mixmaster.sourceforge.net`.

[13] S.J. Murdoch. Hot or not: Revealing hidden services by their clock skew. In *Proceedings of the 13th ACM conference on Computer and communications security*, pages 27–36. ACM, 2006.

[14] Tor. The tor netowrk. `http://www.torproject.org/`.

[15] M.K. Wright, M. Adler, B.N. Levine, and C. Shields. Passive-logging attacks against anonymous communications systems. *ACM Transactions on Information and System Security (TISSEC)*, 11(2):1–34, 2008.

[16] zzz and L. Schimmer. Peer profiling and selection in the i2p anonymous network. In *PET-CON 2009.1.*, TU Dresden, Germany, 03/2009 2009.

Centre de recherche INRIA Nancy – Grand Est
LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)

Centre de recherche INRIA Bordeaux – Sud Ouest : Domaine Universitaire - 351, cours de la Libération - 33405 Talence Cedex
Centre de recherche INRIA Grenoble – Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier
Centre de recherche INRIA Lille – Nord Europe : Parc Scientifique de la Haute Borne - 40, avenue Halley - 59650 Villeneuve d'Ascq
Centre de recherche INRIA Paris – Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex
Centre de recherche INRIA Rennes – Bretagne Atlantique : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex
Centre de recherche INRIA Saclay – Île-de-France : Parc Orsay Université - ZAC des Vignes : 4, rue Jacques Monod - 91893 Orsay Cedex
Centre de recherche INRIA Sophia Antipolis – Méditerranée : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399